FDN SANDBOX: REAL-TIME EXPERIMENTATION AND ANALYSIS OF FDNS

Alexandre St-Onge and Gary Scavone

Computational Acoustic Modeling Lab
Music Technology, Schulich School of Music
McGill University
Montreal, Canada
alexandre.st-onge2@mail.mcgill.ca, gary.scavone@mcgill.ca

ABSTRACT

This work presents `sfFDN`, a modular and real-time-capable C++ library for Feedback Delay Networks (FDNs), together with the companion FDN Sandbox application designed for interactive experimentation, analysis, and parameter optimization. The library implements the canonical FDN as well as several recent extensions, including filter feedback matrices, velvet-noise decorrelation filters, and two-stage graphic equalizers for attenuation and tone correction. The Sandbox application exposes these features through a graphical interface, providing a suite of real-time visualizations, as well as an optimization framework supporting nine algorithms from the `ensmallen` library, with built-in loss functions for both colorless reverberation and room impulse response matching. Both the library and the application are open-source.

1. INTRODUCTION

Artificial reverberation has a rich history dating back to the 1960s, beginning with the work of Manfred Schroeder [1]. Since then, many methods have been proposed to simulate acoustic spaces, each with its own trade-offs. One of the most popular approaches is the Feedback Delay Network (FDN), favored for its high flexibility and relatively low computational cost [2, 3, 4].

However, the large number of parameters that must be chosen for an FDN can make it difficult to achieve perceptually convincing results. Poor parameter choices quickly lead to unpleasant metallic artifacts or unnatural decay profiles in the resulting sound. This tuning task becomes even more challenging when targeting specific reverberation characteristics, such as replicating the acoustic characteristics of a real-world room. In recent years, differentiable digital signal processing (DDSP) [5] techniques have emerged as a powerful approach to tackle this problem. They have proven highly effective for reducing unwanted coloration and automatically matching synthetic reverberators to measured impulse responses. Despite these advances, bridging the gap between research and practical use remains a challenge. Implementing a differentiable FDN for optimization purposes often results in code that is tightly coupled with machine learning frameworks and too computationally heavy for real-time audio applications like video games, virtual reality, or live music production.

Furthermore, recent FDN extensions such as velvet noise filters [6] or filter feedback matrices [7] have been proposed to further improve the perceptual quality of FDNs. Reference implementations of these methods are often made available by their authors, but they are typically implemented in MATLAB, making it difficult to accurately assess their real-time performance and suitability for use in production environments.

This work introduces the `sfFDN` library, a flexible and real-time-capable C++ implementation of the FDN structure and its recent extensions, as well as the FDN Sandbox application, which allows users to experiment with different FDN configurations and parameters in real-time.

2. THE SFFDN LIBRARY

To power the FDN Sandbox application, a custom C++ library called `sfFDN`[1] was developed. Heavily inspired by the FDN Toolbox MATLAB library [8], `sfFDN` was designed to bridge the gap between research and real-time audio processing by being modular, easily extensible to allow new FDN extensions to be integrated, and performant enough to be used in production environments. The FDN structure presented by Prawda et al. [9] was reimplemented in `sfFDN` to demonstrate the performance of the library. Table 1 shows the time taken to process 128 samples of audio for different FDN sizes compared to the implementation in [9]. The measurements were taken on a computer with an Intel Core i9-12900K CPU.

Table 1: Time (in µs) taken to process 128 samples of audio for different FDN implementations.

FDN Size	4	6	8	16	32
sfFDN	3.8	5.6	7.3	15.0	31.2
RTFDN [9]	6.02	9.55	15.0	39.2	112.0

The FDN topology implemented in the library separates the canonical structure into distinct building blocks: input gains, delay lines, feedback matrix, loop filters, output gains, tone correction filters, and direct gain. The library provides classes to implement the common variations of each of these components, such as constant and time-varying input and output gains, and common feedback matrices (Hadamard, Householder, random orthogonal, circulant, nested allpass, etc.), time-varying delay lines, and several types of IIR filters, including the two-stage approach proposed in [10]. More recent FDN extensions are also implemented, such as filter feedback matrices [7] and velvet noise filters [6]. Figure 1 shows the architecture of the FDN implementation in

[1] `https://github.com/Segfault1602/sfFDN`

`sfFDN`. New components can easily be added by inheriting from the `AudioProcessor` base class.

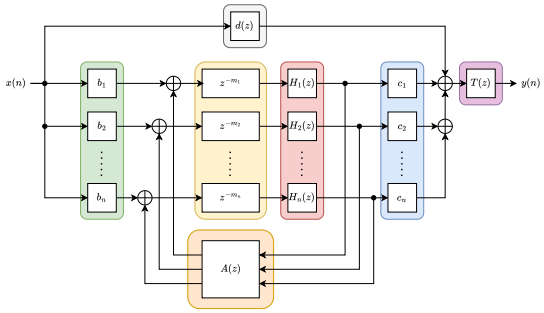


Figure 1: Architecture of the FDN implementation.

3. FDN SANDBOX APPLICATION

The FDN Sandbox application[2] allows users to experiment with different FDN configurations and parameters in real-time. Figure 2 shows a screenshot of the main interface of the application.

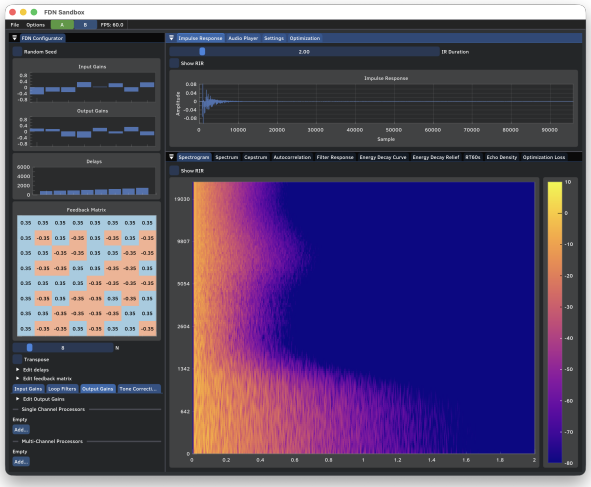


Figure 2: Main interface of the FDN Sandbox application.

The application includes an audio player enabling users to listen to the generated impulse responses and auralize reverberated sound files. It also features a convolution reverb module to allow users to auralize external impulse responses and directly A/B compare them to the synthesized FDN sound.

3.1. Real-time Visualizations

Since the application serves as a tool for experimentation and research, it encompasses a wide variety of graphs that can be valuable when analyzing FDNs. These plots, which update dynamically as the user modifies the FDN's parameters, include:

- Spectrogram and Mel-spectrogram
- Modal excitation distribution
- Spectrum and Cepstrum
- Time and spectral-domain autocorrelation
- Frequency and phase response of the attenuation filters
- Energy decay curve (EDC) and Mel-scale energy decay relief (EDR)
- Reverberation time (RT_{60})
- Echo density profile

3.2. Optimization Interface

The application also incorporates a graphical interface to automatically optimize certain parameters of the FDN. The optimization is performed directly on the `sfFDN` FDN, bypassing the need to use a differentiable implementation. The framework supports two primary variants of optimizations: optimization for colorless reverberation and optimization for room impulse response (RIR) matching. Nine optimization algorithms from the `ensmallen` optimization library [11], summarized in Table 2, are currently supported. For the algorithms that require gradient information, the gradients are computed using finite differences; both forward and central finite difference methods are implemented and can be selected by the user through the interface.

Table 2: Optimization algorithms available in the FDN Sandbox.

Gradient-Based	Gradient-Free
Gradient Descent	SPSA
Adam	Simulated Annealing
L-BFGS	CNE
	Differential Evolution
	Particle Swarm Optimization
	CMA-ES

3.2.1. Optimization for Colorless Reverberation

The input/output gains and the orthogonal matrix coefficients are fine-tuned to minimize modal coloration. Three types of matrices are currently supported for optimization: random orthogonal, random Householder, and circulant matrix. For an FDN of size N, the total number of parameters to optimize is $N^2 + 2N$: N^2 for the random orthogonal matrix and $2N$ for the input and output gains. If a Householder or circulant matrix is used, the number of parameters needed to define the feedback matrix is reduced to N, which can significantly speed up the optimization process for larger FDN sizes.

Two loss functions are implemented for the optimization of the FDN parameters for colorless reverberation: a spectral flatness loss, and a time-domain sparsity loss. The spectral flatness loss is inspired by the spectral loss function used in [12] and aims to maximize the spectral flatness of the generated impulse response. Spectral flatness is defined as the ratio between the geometric mean and the arithmetic mean of the power spectrum and is given by the equation:

$$\text{Flatness} = \frac{\sqrt[K]{\prod_{k=0}^{K-1} X(k)}}{\frac{1}{K}\sum_{k=0}^{K-1} X(k)}, \tag{1}$$

[2] `https://github.com/Segfault1602/FDNSandbox`

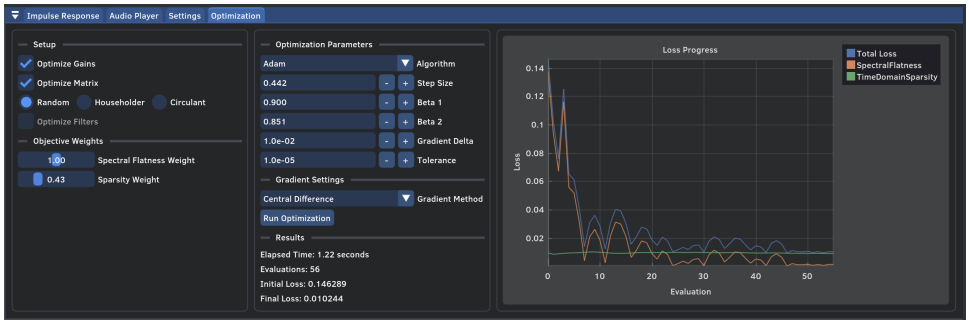


Figure 3: Optimization interface in the FDN Sandbox.

where $X(k)$ is the power spectrum at bin k [13]. The spectral flatness is a measure of how noise-like a signal is, with a value approaching 1 for a perfectly flat spectrum and a value of 0 for a pure tone. Since the optimization framework used in this work expects a loss function to be minimized, the spectral flatness loss $\mathcal{L}_{\text{flatness}}$ is defined as:

$$\mathcal{L}_{\text{flatness}} = |\text{Flatness}(X) - 0.56|, \tag{2}$$

where X is the power spectrum of the generated impulse response. The constant 0.56 was found empirically, and corresponds to the average spectral flatness of randomly generated noise signals of length $L = 48000$ samples, which is consistent with the findings of [14]. As mentioned in [12], the spectral flatness loss can lead to optimization results that have a very low echo density. To mitigate this issue, the time-domain sparsity loss defined in [12] was also implemented and is defined as follows:

$$\mathcal{L}_{\text{sparsity}} = \frac{\|x\|_2}{\|x\|_1}, \tag{3}$$

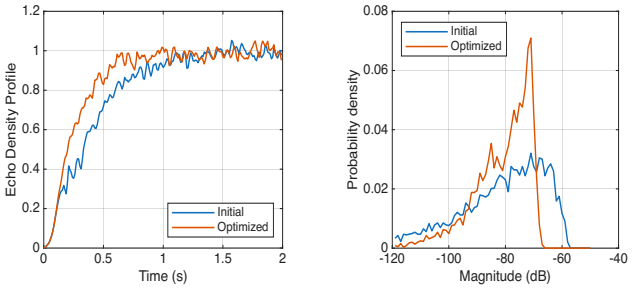
where x is the impulse response and the operator $\|\cdot\|_p$ is the ℓ_p norm. The total loss function is then defined as:

$$\mathcal{L}_{\text{color}} = \alpha_1 \mathcal{L}_{\text{flatness}} + \alpha_2 \mathcal{L}_{\text{sparsity}}, \tag{4}$$

where α_1 and α_2 are weighting factors used to balance the contribution of each loss term and can be adjusted by the user to prioritize one loss over the other. Figure 4 shows the effect of this optimization process on a 6-channel FDN. Figure 4a shows a slight improvement, seen as a faster buildup of the echo density in the optimized FDN compared to the initial FDN. Figure 4b shows a narrower distribution of the modal excitation in the optimized FDN, which correlates with a reduction in coloration [15].

3.2.2. Optimization for Room Impulse Response Matching

For this step, the parameters optimized are the frequency-dependent $RT_{60}(\omega)$ of the attenuation filters, the gains $T(\omega_i)$ of the tone correction filter, and an additional global gain factor G_{global}. The filter design used for this optimization is the octave band filter design proposed in [10]. As the filter used gives control over 10 frequency bands, the number of parameters to optimize is 10 for the RT_{60} values, 10 for the tone correction filter gains, and 1 for the global gain factor, for a total of 21 parameters. Two loss functions from [16, 17] are currently implemented for this step, $\mathcal{L}_{\text{EDR}}$ and $\mathcal{L}_{\text{EDC}}$,



(a) **(b)**

Figure 4: Echo density profile (a) and modal excitation distribution (b) of the optimized FDN compared to the initial FDN.

defined as follows:

$$\mathcal{L}_{\text{EDR}} = \frac{\sum_k \sum_m \left|\text{EDR}_{\text{mel}}^{\text{target}}(m,k) - \text{EDR}_{\text{mel}}^{\text{optim}}(m,k)\right|}{\sum_k \sum_m \left|\text{EDR}_{\text{mel}}^{\text{target}}(m,k)\right|}, \tag{5}$$

$$\mathcal{L}_{\text{EDC}} = \frac{\sum_n \left(\text{EDC}^{\text{target}}(n) - \text{EDC}^{\text{optim}}(n)\right)^2}{\sum_n \text{EDC}^{\text{target}}(n)^2}, \tag{6}$$

where $\text{EDR}_{\text{mel}}^{\text{target}}$ and $\text{EDR}_{\text{mel}}^{\text{optim}}$ are the Mel-scale EDRs of the target and optimized impulse responses, respectively, and $\text{EDC}^{\text{target}}$ and $\text{EDC}^{\text{optim}}$ are the EDCs of the target and optimized impulse responses, respectively. The total loss function is then defined as

$$\mathcal{L}_{\text{spectrum}} = \alpha_1 \mathcal{L}_{\text{EDR}} + \alpha_2 \mathcal{L}_{\text{EDC}}, \tag{7}$$

where α_1 and α_2 are weighting factors that can be adjusted by the user to prioritize one loss over the other. Early reflections can be taken into account in the optimization process by including an FIR filter in the direct path of the FDN during the optimization. Figure 5 shows the EDR of the optimized impulse response compared to the target RIR and to the impulse response obtained through the analysis-synthesis method described in [18]. The optimization process successfully matches the EDR of the target impulse response. While the analysis-synthesis method is able to accurately match the slope of the EDR, it struggles to match the initial energy levels of the EDR, which results in the offset between the two EDR curves seen in Figure 5. The target RIR used was generated using a mix of image-source and ray-tracing methods using the `pyroomacoustics` library [19] and the corresponding early

reflections filter was generated using the same method with a maximum reflection order of 3.

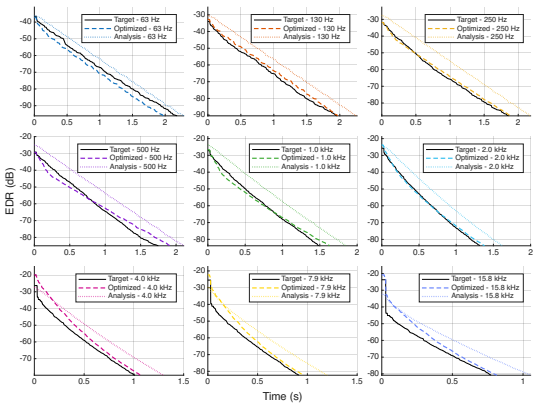


Figure 5: EDR of the optimized impulse response compared to the target and to the impulse response obtained through the RIR2FDN analysis-synthesis method [18].

4. CONCLUSIONS

This work introduced the `sfFDN` library, a flexible and real-time-capable C++ implementation of the FDN structure and some of its recent extensions, and the FDN Sandbox application, which allows users to experiment with different FDN configurations and parameters in real-time. An optimization framework is also included and results show that it can effectively reduce coloration and match target RIRs without requiring a differentiable FDN implementation. Both the `sfFDN` library and the FDN Sandbox application are open-source and available online.

5. REFERENCES

[1] M. R. Schroeder and B. F. Logan, ""Colorless" Artificial Reverberation," *The Journal of the Acoustical Society of America*, vol. 9, no. 3, pp. 192–197, Jul. 1961.

[2] M. Gerzon, "Synthetic Stereo Reverberation, Part II," *Studio Sound*, vol. 14, pp. 24–28, 1972.

[3] J. Stautner and M. Puckette, "Designing Multi-Channel Reverberators," *Computer Music Journal*, vol. 6, no. 1, pp. 52–65, 1982.

[4] J.-M. Jot and A. Chaigne, "Digital Delay Networks for Designing Artificial Reverberators," in *Proceedings of the 90th Audio Engineering Society Convention (AES)*, no. 3030, Paris, France, Feb. 1991, pp. 1–16.

[5] B. Hayes, J. Shier, G. Fazekas, A. McPherson, and C. Saitis, "A review of differentiable digital signal processing for music and speech synthesis," *Frontiers in Signal Processing*, vol. 3, p. 1284100, Jan. 2024.

[6] J. Fagerström, B. Alary, S. J. Schlecht, and V. Välimäki, "Velvet-Noise Feedback Delay Network," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020)*, Vienna, Austria, 2020, pp. 219–226.

[7] S. J. Schlecht and E. A. P. Habets, "Scattering in Feedback Delay Networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1–11, Jun. 2020.

[8] S. J. Schlecht, "FDNTB: The Feedback Delay Network Toolbox," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, Vienna, Austria, 2020, pp. 211–218.

[9] K. Prawda, V. Välimäki, S. Willemsen, and S. Serafin, "Flexible Real-Time Reverberation Synthesis with Accurate Parameter Control," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, Vienna, Austria, 2020, pp. 16–23.

[10] V. Välimäki, K. Prawda, and S. J. Schlecht, "Two-Stage Attenuation Filter for Artificial Reverberation," *IEEE Signal Processing Letters*, vol. 31, pp. 391–395, Jan. 2024.

[11] R. R. Curtin, M. Edel, R. G. Prabhu, S. Basak, Z. Lou, and C. Sanderson, "The Ensmallen Library for Flexible Numerical Optimization," *Journal of Machine Learning Research*, vol. 22, no. 166, pp. 1–6, 2021, arXiv:2108.12981 [cs].

[12] G. Dal Santo, K. Prawda, S. J. Schlecht, and V. Välimäki, "Differentiable Feedback Delay Network for Colorless Reverberation," in *Proceedings of the 26th International Conference on Digital Audio Effects (DAFx23)*, Copenhagen, Denmark, Sep. 2023, pp. 244–251.

[13] J. Johnston, "Transform coding of audio signals using perceptual noise criteria," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 2, pp. 314–323, Feb. 1988.

[14] N. Agus, H. Anderson, J.-M. Chen, S. Lui, and D. Herremans, "Perceptual Evaluation of Measures of Spectral Variance," *The Journal of the Acoustical Society of America*, vol. 143, no. 6, pp. 3300–3311, Jun. 2018.

[15] J. Heldmann and S. J. Schlecht, "The Role of Modal Excitation in Colorless Reverberation," in *Proceedings of the 24th International Conference on Digital Audio Effects (DAFx20in21)*. Vienna, Austria: IEEE, Sep. 2021, pp. 206–213.

[16] A. I. Mezza, R. Giampiccolo, and A. Bernardini, "Modeling the Frequency-Dependent Sound Energy Decay of Acoustic Environments with Differentiable Feedback Delay Networks," in *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx24)*, Guildford, Surrey, UK, 2024, pp. 238–245.

[17] A. I. Mezza, R. Giampiccolo, E. De Sena, and A. Bernardini, "Data-Driven Room Acoustic Modeling Via Differentiable Feedback Delay Networks with Learnable Delay Lines," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2024, no. 1, p. 51, Oct. 2024.

[18] G. Dal Santo, B. Alary, K. Prawda, S. Schlecht, and V. Välimäki, "RIR2FDN: An Improved Room Impulse Response Analysis and Synthesis," in *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx24)*, Guildford, Surrey, UK, Sep. 2024, pp. 230–237.

[19] R. Scheibler, E. Bezzam, and I. Dokmanić, "Pyroomacoustics: a python package for audio room simulations and array processing algorithms," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, pp. 351–355.